CRT-2001-23

Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation

A GUIDE TO VEHICLE ROUTING HEURISTICS

by

Jean-François Cordeau
Michel Gendreau
Gilbert Laporte
Jean-Yves Potvin
Frédéric Semet

June 2001

102-02943/

PUBLICATION
CRT-2001-23

A GUIDE TO VEHICLE ROUTING HEURISTICS*

by

Jean-François Cordeau[1,2]
Michel Gendreau[2,3]
Gilbert Laporte[1,2]
Jean-Yves Potvin[2,3]
Frédéric Semet[4]

*This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR).

[1] Chaire de distributique, École des Hautes Études Commerciales, 3000, chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7
[2] Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3T 3J7
[3] Département d'informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7
[4] LAMIH-ROI, Université de Valenciennes et du Hainaut Cambresis, Le Mont Houy, 59313 Valenciennes Cedex 9, France

Département d'informatique et de recherche opérationnelle - Publication #1196

Dépôt légal – Bibliothèque nationale du Québec
Bibliothèque nationale du Canada, 2001

ABSTRACT

Several of the most important classical and modern heuristics for the Vehicle Routing Problem are summarized and compared using four criteria : accuracy, speed, simplicity and flexibility. Computational results are reported.

Key words : vehicle routing problem, heuristics.

RÉSUMÉ

On compare plusieurs heuristiques classiques et modernes pour le problème de tournées de véhicules. Quatre critères sont utilisés dans la comparaison : l'exactitude de la solution, la vitesse de l'algorithme, sa simplicité et sa flexibilité. On présente des résultats numériques.

Mots-clefs : problème de tournées de véhicules, heuristiques.

Introduction

The *Vehicle Routing Problem* (VRP), introduced by Dantzig and Ramser[1] in 1959, holds a central place in distribution management and has become one of the most widely studied problems in combinatorial optimization. The *Classical VRP* can be formally defined as follows. Let $G = (V, A)$ be a graph where $V = \{v_0, v_1, ..., v_n\}$ is a vertex set, and $A = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ is an arc set. Vertex v_0 represents a depot, while the remaining vertices correspond to customers. With A are associated a cost matrix (c_{ij}) and a travel time matrix (t_{ij}). If these matrices are symmetrical, as is commonly the case, then it is standard to define the VRP on an undirected graph $G = (V, E)$, where $E = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$ is an edge set. Each customer has a non-negative demand q_i and a service time t_i. A fleet of m identical vehicles of capacity Q are based at the depot. The number of vehicles is either known in advance or treated as a decision variable. The VRP consists of designing a set of at most m delivery or collection routes such that 1) each route starts and ends at the depot, 2) each customer is visited exactly once by exactly one vehicle, 3) the total demand of each route does not exceed Q, 4) the total duration of each route (including travel and service times) does not exceed a preset limit D, and 5) the total routing cost is minimized. A common variant is where a time window $[a_i, b_i]$ is imposed on the visit of each customer. Several other extensions have also been studied: the vehicle fleet may be heterogeneous[2], vehicles may perform both collections and deliveries on the same route[3], some vehicles may be unable to visit certain sites[4], some customers may require several visits over a given time period[5], there may exist more than one depot[5], deliveries may be split among several vehicles[6], etc. For overview articles on the VRP, see Golden and Assad[7], Fisher[8], Desrosiers et al.[9], Crainic and Laporte[10], Toth and Vigo[11], Laporte and Semet[12], Gendreau, Laporte and Potvin[13], and Cordeau et al.[14].

The VRP is a hard combinatorial optimization problem and only relatively small instances can be solved to optimality. In fact no exact algorithm is capable of consistently solving instances in excess of 50 customers[15]. This is due to the fact that sharp

lower bounds on the objective function value are hard to derive, which means that partial enumeration based exact algorithms (using branch-and-bound or dynamic programming) will have a slow convergence rate. Since exact approaches are in general inadequate, heuristics are commonly used in practice.

There has been a steady evolution, over the past forty years, in the development of VRP heuristics. In the early *classical heuristics*, much of the emphasis was put on quickly obtaining a feasible solution and possibly applying to it a postoptimization procedure. This class of methods includes the well-known savings algorithm[16], the sweep algorithm[17], and the Fisher and Jaikumar algorithm[18]. Over the last ten years, much of the research effort has concentrated on the development of algorithms based on *metaheuristics*, using mainly two principles: *local search* and *population search*. In local search methods, an intensive exploration of the solution space is performed by moving at each step from the current solution to another promising solution in its neighbourhood. *Simulated annealing* (SA)[19] and *tabu search* (TS)[20]are two prime examples of this principle. Population search consists of maintaining a pool of good parent solutions and recombining them to produce offspring. A classical example is *genetic search* (GS)[21] which combines two parents to produce offspring. *Adaptive memory procedures* (AMPs)[22] can be viewed as an extension of GS where several parents are used to produce several offspring. While more time consuming than the early heuristics, metaheuristics are capable of consistently producing high quality solutions. Thus, on a set of fourteen classical benchmark instances[23], the most powerful methods have produced one proven optimum while the solutions obtained on the remaining instances are believed to be optimal or near-optimal since they have not been improved after thousands of hours of computing effort, using sophisticated search techniques.

Yet most commercial software and several in-house computer programs used by companies are based on unsophisticated methodologies dating sometimes from the 1960s. There are several reasons for this state of affairs. One is that the optimization

component of VRP software is only a small part of the product, most of the effort being expanded on data management and sophisticated user interfaces. It is not uncommon for routing experts to use low quality solutions produced by VRP software and exploit the interactive capabilities of the system to perform manual improvements. Another reason is that company analysts and software developers are simply unaware of the latest algorithmic developments. While there is some truth to this, we believe the main hindrance to technology transfer may be more deep-rooted. In this article, we argue that most of the available VRP heuristics simply lack some of the necessary attributes to ensure their adoption by practitioners.

In the following section we describe what we believe are four essential attributes for software transferability and end-user adoption. We then provide an appraisal of the best known heuristics with respect to these criteria. As much as we have tried to base our analysis on published and objective data, our analysis is at times personal and critical, especially with respect to the more intangible criteria. The algorithms are grouped in two categories: classical heuristics and metaheuristics. A summary and conclusion close the article.

Four attributes of good VRP heuristics

Vehicle routing heuristics, as are most heuristics, are usually measured against two criteria: *accuracy* and *speed*. In our opinion *simplicity* and *flexibility* are also essential attributes of good heuristics. We now elaborate on these four criteria.

Accuracy

Accuracy measures the degree of departure of a heuristic solution value from the optimal value. Since optima and sharp lower bounds are usually unavailable in the case of the VRP, most comparisons have to be made with best known values. As pointed out by Barr et al.[24], analyzing heuristic results is fraught with difficulties. Authors often report results obtained for the best combination of algorithmic parameters, or for the best of several runs with different starting solutions. Not all

authors use the same rounding or truncating conventions which, as emphasized by Gendreau, Hertz and Laporte[25], can lead to vastly different results. For example, on the first instance of the Christofides, Mingozzi and Toth (CMT)[23] test problems, the same tabu search algorithm (Taburoute) yields a cost of 524.61 if computations are performed with real costs c_{ij} with five significant digits (this is, by the way, a proven optimum[26]), a cost of 521 if costs are rounded up or down to the nearest integer, and a cost of 508 if truncated costs are used. Thus integer results reported without mention of the rounding rule are to a large extent meaningless. For a discussion on the bias introduced by rounding, see Mole[27].

Another issue related with accuracy is consistency. As a rule, users will prefer a heuristic that performs well all the time rather than one that may perform even better most of time but very poorly on other occasions, and may even produce solutions easily perfectible by visual inspection. Such solutions are enough to discredit the algorithm.

Finally, users will often prefer an algorithm that produces a good solution at an early stage, and then displays solutions of increasing quality throughout the execution to an algorithm that comes up with only a final answer, possibly after a long computing time. This gives users a better feel of how much additional effort is worth investing given the evolution rate of the solution value.

Speed

Just how important is computation speed in vehicle routing? It all depends on the planning level at which the problem is solved and on the degree of accuracy required. At one extreme, real-time applications such as express courier pickup and delivery[28] or ambulance redeployment[29] require fast, sometimes almost instantaneous action. For example, the paper by Gendreau, Laporte and Semet[29] describes the crucial role played by parallel computing in a setting where an ambulance relocation strategy must be determined every three minutes on the average. At the other extreme, in

long term planning decisions made every several months, such as fleet sizing, it makes sense to invest several hours or even several days of computing time, particularly if large sums of money are at stake. Most applications fall somewhere between these two extremes. It does not seem unreasonable to invest ten or twenty minutes of computing time on a routing problem that must be solved daily. Interactive systems must of course react much more quickly. The issue of speed is not always properly put into perspective. For example, wouldn't most practitioners prefer a VRP heuristic that is 2.38% accurate and runs in 3.48 seconds to one that is 5.85% accurate but requires only 0.26 second of computing time? (These are actual statistics of the 2-petal and 1-petal algorithms described by Renaud, Boctor and Laporte[30].)

Accurate reporting of computing time is another delicate issue in the scientific literature, particularly in the case of multiple runs or if parallel computing is used. As in the area of accuracy, strict standards are not consistenly enforced by VRP researchers when it comes to assessing the speed of algorithms.

Simplicity

Several VRP heuristics are never implemented because they are just too complicated to understand and to code. While it is unrealistic to expect scientific articles to provide a minute description of every algorithmic detail, sufficient information should be provided to enable a reasonably skilled programmer to come up with a working code. In addition, heuristics should be reasonably robust to ensure that they work properly even if not every single detail is implemented. Many algorithmic descriptions fail on the count of providing too much or insufficient detail. One reason why the Clarke and Wright[16] algorithm is so popular among practitioners is that its basic principle is easy to understand and trivial to code. (Amazingly, the original description is rather clumsy and would probably fall short of today's publishing standards.) Simple codes (preferably short and self-contained) stand a better chance of being adopted, although a minimum of complexity is to be expected for good results.

Algorithms that contain too many parameters are difficult to understand and unlikely to be adopted. This problem is prevalent in most metaheuristics developed over the past ten years. In their quest for ever better solutions, researchers have increased the number of parameters contained in their algorithms far beyond what can be deemed reasonable, particularly on view of the fact that relatively few instances are used in the tests. This problem was recently raised by Golden et al.[31]. Not only should the number of algorithmic parameters be limited, but these should also make sense to the end-user. Thus a parameter controlling the number of iterations without improvements in a local search heuristic is easy to understand, whereas a bound on the number of executions of an internal technical procedure is meaningless to most people.

There are two easy ways round the proliferation of parameters. One is to set them once and for all at some meaningful value, especially if tests show that the algorithm is rather insensitive to a particular parameter choice. Another possibility is to make use of parameters that self-adjust during the course of the algorithm[5,25].

Flexibility

A good VRP heuristic should be flexible enough to accomodate the various side constraints encountered in most real-life applications. While most of the VRP literature focuses on capacity and sometimes route length constraints, it is often clear how changes can be made to deal with additional constraints, but this is not always possible, and performance can also worsen significantly as a result. Our experience[5,25] suggests that an efficient way of handling side constraints in a local search process is through the use of two objectives. The first, $F(x)$, computes the routing cost of solution x. The second, $F'(x)$, is the sum of $F(x)$ and weighted penalty terms associated with violations of each side constraint. For example, if $Q(x)$ and $D(x)$ are the capacity and route duration violations associated with solution x, then $F'(x)$ would be defined as $F(x)+\alpha Q(x)+\beta D(x)$, where α and β are positive self-adjusting penalty parameters. Initially set equal to 1, these parameters are periodically increased or

decreased throughout the search according to whether previous solutions were infeasible or feasible. This way of proceeding means that the search will probably evolve through a mix of feasible and infeasible solutions, thus reducing the probability of becoming trapped in a local minimum. Another algorithmic advantage of this device is that the search can operate with relatively simple moves, such as removing a customer from its current route and inserting it in a different route. When feasibility must be maintained at all cost, such simple moves tend not to work if side constraints are tight, and more complex and time consuming operations must then be envisaged, such as ejection chains[32]. So in a sense, algorithmic flexibility is in part achieved through simplicity of design.

Classical heuristics

Several heuristics have been devised for the VRP[12], only some of which are sufficently well known to be truly viewed as "classical". For the sake of parsimony we concentrate on three of the best know heuristics: the Clarke and Wright algorithm, the sweep algorithm and the Fisher and Jaikumar algorithm. These were selected partly because of their popularity, and also because they operate on vastly different principles. Some extensions of these methods are also discussed in passing.

The Clarke and Wright savings heuristic

The Clarke and Wright (CW)[16] heuristic is one of the best known and remains to this day widely used in practice, despite some of its shortcomings. It is based on the notion of *saving*. Initially, a feasible solution consists of n back and forth routes between the depot and a customer. At any given iteration, two routes $(v_0, ..., v_i, v_0)$ and $(v_0, v_j, ..., v_0)$ are merged into a single route $(v_0, ..., v_i, v_j, ..., v_0)$ whenever this is feasible, thus generating a saving $s_{ij} = c_{i0} + c_{0j} - c_{ij}$. In the *parallel* version of the algorithm, the merge yielding the largest saving is always implemented, whereas the *sequential* version keeps expanding the same route until this is no longer feasible. In practice the parallel version is much better. It is customary to apply a 3-opt

post-optimization step to the final solution.

This algorithm scores very high on simplicity and speed. It contains no parameters and it is easy to code. In our tests performed on the CMT benchmark instances, it typically ran within 0.12 second on a Sun Ultrasparc 10 workstation (440 MHz), without the 3-opt step. Executing a post-optimization increased the average running time to only 0.13 second. This algorithm obtains a medium score on accuracy. The best implementation (parallel version followed by 3-opt) produced an average deviation of 6.71% from the best known solution values identified by Taillard[33] and by Rochat and Taillard[22] (see Table 1). In addition, several researchers have observed that the solution is sometimes of rather poor quality and often contains at least one rather circumferential route. The lack of flexibility is probably the worst feature of this algorithm. While additional constraints can in principle be incorporated in the CW algorithm, this usually results in a sharp deterioration in solution quality. This can be explained by the fact that the algorithm is greedy by nature and contains no mechanism to undo early unsatisfactory route merges. Solomon[34] reports a variant of the CW algorithm in which savings are adapted to handle time windows, but results are not excellent.

Several improvements to the CW algorithm have been proposed. Gaskell[35] and Yellow[36] have suggested using generalized savings of the form $s_{ij} = c_{0i} + c_{j0} - \lambda c_{ij}$ to help produce more compact routes, where $\lambda > 0$. Other enhancements concern the use of sophisticated data structures and sorting strategies[37,38] to better handle the savings. We believe, however, that given the present level of computer technology and the very fast running time of the CW algorithm on medium size instances, these technical improvements are probably no longer relevant.

Another stream of research on the CW algorithm has concentrated on optimizing the route merging process through the use of a matching algorithm. Early results based on this idea were produced by Desrochers and Verhoog[39] and by Altinkemer and Gavish[40]. The best and most recent implementation is due to Wark and Holt[41]

who often obtained significant improvements over the original CW implementation, but at the expense of much increased computing times. Their results, presented in Table 1, correspond to the best of five runs, each run requiring on the average between 4 and 107 minutes on a Sun4/630MP. On the whole, this algorithmic enhancement removes from the CW heuristic two of its best features (speed and simplicity), is heavy to implement and does nothing to redress the lack of flexibility of the original algorithm. While the repeated application of a matching based algorithm yields increased accuracy (it produces average deviation of 0.63% on the 14 CMT benchmark instances), this technique has in our opinion limited potential because of its complexity and low level of flexibility.

The sweep algorithm

The sweep algorithm is generally attributed to Gillett and Miller[17] although its principle can be traced back to Wren[42] and Wren and Holliday[43]. It applies to planar instances of the VRP. Feasible routes are created by rotating a ray centered at the depot and gradually including customers in a vehicle route until the capacity or route length constraint is attained. A new route is then initiated and the process is repeated until the entire plane has been swept. A 3-opt step is then typically applied. The Renaud, Boctor and Laporte[30] implementation of this algorithm on the 14 CMT instances has yielded an average deviation of 7.09% from the best known solution values. Average computing times were 1.76 seconds on a Sun Sparcstation 2 (210.5 Mips, 4.2 Mflops), with 32 megabytes of RAM (see Table 1). Again, this algorithm scores high on simplicity, but it does not seem to be superior to CW both in terms of accuracy and speed. It is also rather inflexible. Again, the greedy nature of the sweep mechanism makes it difficult to accomodate extra constraints and the fact that the algorithm assumes a planar structure severely limits its applicability. In particular, it is not well suited to instances defined in an urban setting with a grid street layout.

A number of heuristics generate feasible vehicle routes (sometimes called *petals* in this context) and determine a best combination through the solution of a set

partitioning problem. Prime examples of this approach are the 1-petal algorithms of Foster and Ryan[44] and of Ryan, Hjorring and Glover[45], and the 2-petal heuristic of Renaud, Boctor and Laporte[30] where, in addition to single routes, double vehicle routes are also generated. Both these extensions provide accuracy gains with respect to the sweep algorithm. In terms of speed, improvements can also be made since the 3-opt step is no longer applied (see Table 1). These extensions do not get full marks on simplicity since generating a large pool of petals (especially 2-petals) can be cumbersome, and a set partitioning step must be added. As far as flexibility is concerned, petal algorithms can be made to accomodate a wide variety of constraints but this can come at the expense of simplicity. In fact this type of algorithm can be viewed as a truncated version of column generation which is known to produce high quality results on tightly constrained VRPs with time windows[46], but again simplicity is sacrificed.

The Fisher and Jaikumar algorithm

The Fisher and Jaikumar[18] algorithm is a two-phase process in which feasible clusters of customers are first created by solving a generalized assignement problem (GAP), and a vehicle route is determined on each cluster by means of a traveling salesman problem (TSP) algorithm. To formulate the GAP, it is necessary to first determine a *seed* for each route from which customer distances are computed. Since the GAP is NP-hard, it is usually solved by means of a Lagrangean relaxation technique. While good results were reported for this algorithm in the early 1980s we have several misgivings about its performance. The original article[18] provides integer solutions values which, as was mentioned later[8], were obtained with c_{ij} coefficients rounded to the nearest integer, and the solutions cannot be verified, which makes the assesssment of this algorithm difficult. Even disregarding this, solution values reported by Fisher and Jaikumar are sometimes relatively far from the best available for rounded c_{ij}s. The speed of this algorithm is also difficult to assess and highly related to the choice of seeds and to the convergence rate of the Lagrangean process

(this is at least our experience with our implementation). The lack of simplicity is one major problem of the Fisher and Jaikumar algorithm. For a start, we found that if seeds are selected as in Fisher et al.[47] and the Lagrangean steps programmed as recommended in Fisher, Jaikumar and Van Wassenhove[48], then convergence is often poor, several trials may be necessary to reach a satisfactory solution, and even then accuracy can be low. We did not succeed in obtaining results close to those of Fisher and Jaikumar[18]. Flexibility is also problematic. While it is in principle simple to handle capacity constraints, the original reference contains no indication regarding the treatment of route length constraints, although six instances involving such constraints are reportedly solved. We see no easy mechanism by which to incorporate these or other side constraints within the GAP algorithm. Bramel and Simchi-Levi[49] have optimized the choice of seeds in the Fisher and Jaikumar algorithm by solving a capacitated location problem. Their results on the seven CMT instances containing only capacity constraints show a relatively large average deviation (3.29%) from the best known results (see Table 2).

Metaheuristics

With respect to classical heuristics, metaheuristics perform a much more thorough search of the solution space, allowing worsening and sometimes infeasible moves, as well as recombinations of solutions to create new ones. This area of research has known a formidable growth over the past ten years and has produced some highly effective and flexible VRP heuristics[13]. It is fair to say, however, that the gains in solution quality obtained with these modern heuristics have often been made at the expense of speed and simplicity, although this is not so true of some of the most recent implementations. The past decade has been very rich in algorithmic design and testing. Some promising ideas did not translate into equally good algorithms while others have withstood the test of time.

Tabu search (TS) clearly stands out as the best metaheuristic for the VRP. By

and large, the best TS implementations for the VRP dominate other search processes such as simulated[50] and deterministic[51,52] annealing, genetic search[53], ant systems[54] and neural networks[55] (see, e.g., Gendreau, Laporte and Potvin[13]). The idea behind TS is to perform a local search by moving, at iteration t, from a solution x_t to a solution x_{t+1} in its neighbourhood. Since moving from x_t to x_{t+1} may cause the objective function to deteriorate, an anti-cycling mechanism is put in place, namely any solution possessing some attribute of x_t is declared *tabu*, or forbidden, for a number of iterations. So, in effect, the best neighbour x_{t+1} of x_t is only selected if it is non-tabu or if it is clear cycling will not occur. Several additional mechanisms such as diversification and intensification have been implemented by a variety of researchers. Of course not all TS heuristics for the VRP have been equally successful. The first known implementation, by Willard[56], did not make use of a sufficiently powerful neighbourhood structure to allow the identification of high quality solutions. In contrast, several implementations that followed contained too many devices and user-controlled parameters. There has been a tendency in recent years toward leaner implementations. In what follows we present some of the best available TS heuristics for the VRP.

Taburoute

With respect to the earlier TS implementations, the Taburoute heuristic of Gendreau, Hertz and Laporte[25] is rather involved and contains several innovative features. The neighbourhood of x_t is the set of all solutions reachable from x_t by removing a vertex v from its current route r and inserting it in another route s containing one of its p closest neighbours by means of a generalized insertion (GENI) procedure[57]. Reinserting v in route r is then declared tabu for θ iterations, where θ is randomly drawn from the interval $[5, 10]$, as suggested by Taillard[58] in the context of the quadratic assignment problem. Intermediate infeasible solutions are considered and penalized through the use of self-adjusting penalty parameters, as explained earlier. Periodic reoptimization of the individual vehicle routes are performed by means of the GENIUS

heuristic[57] for the *Traveling Salesman Problem* (TSP). A continuous diversification strategy is also applied to penalize frequently moved vertices. This is done by adding to the objective function a term proportional to the absolute frequency of movement of the vertex v currently being considered. False starts are also employed: this allows a limited search starting from several initial solutions, and a full search is then performed using the most promising starting point. As is commonly done, the procedure ends when the objective has not improved for a number of consecutive iterations.

On the CMT test problems, Taburoute has produced high quality solutions: the average deviation from the best known values is 0.73% and five best known solutions were produced. Running times can be described as good but not excellent (solution times vary between 6 and 100 minutes on Silicon Graphics workstation (36 MHz, 5.7 Mflops)). Taburoute probably scores rather low on simplicity, partly due to its large number of user-controlled parameters (nine in all) and to its use of the outside TSP heuristic GENIUS which is not that easy to reproduce. Flexibility, however, is relatively high given that additional constraints can easily be incorporated through the penalty mechanism.

The Taillard tabu search algorithm

The Taillard[33] TS implementation was developed at the same time as Taburoute. It also uses random tabu durations and continuous diversification. Where it differs most from Taburoute is in the neighbourhood structure. Standard vertex insertions and exchanges are used instead of GENI. Periodic route reoptimizations are performed by means of an exact TSP algorithm. To help speed up computations, Taillard partitions the problem into several subproblems, each of which is solved independently on a parallel processor. In the case of planar problems, the decomposition process uses concentric rings carved into sectors centered at the depot. For non-planar problems a different decomposition method based on computation of shortest arborescences is used. The boundaries of the subproblems are redefined dynamically.

Taillard's algorithm is one of the best available in terms of accuracy. It has identified twelve of the fourteen best known results on the CMT instances. Computation times are not reported and are difficult to establish with any degree of accuracy. The algorithm is, in one respect at least, simpler than Taburoute since it uses standard insertions, but managing the dynamic decomposition process as well as the parallel implementation adds to its complexity. One would expect this algorithm to reasonably handle additional side contraints because of the combination of insertion and exchange moves used to define neighbour solutions. We believe, however, that resorting to simple insertions and a penalized objective function should offer more flexibility.

The adaptive memory procedure of Rochat and Taillard

The concept of adaptive memory, developed by Rochat and Taillard[22], is probably one of the most powerful ideas put forward in the area of metaheuristics in recent years. An adaptive memory is a pool of good solutions produced by a heuristic, which is dynamically updated by adding to it new elements and removing from it some of its least interesting elements. New elements are generated by recombining good solutions from the pool. The memory update process can therefore be viewed as a form of population search and as a generalization of genetic search. Rochat and Taillard[22] have developed an adaptive memory mechanism for the capacity and distance constrained VRP and for the VRP with time windows, based on the earlier TS algorithms of Taillard[33] and of Rochat and Semet[59]. Given a pool of good VRP solutions, new solutions are obtained by extracting high quality vehicle routes, where routes belonging to better solutions are given a higher probability of being extracted. While extracting vehicle routes care is taken not to include routes that contain already covered customers. Eventually this process will stop with a set of selected routes and some unrouted customers. A new solution is then reconstituted from these routes and unrouted customers using TS. It is then included in the pool if it is of sufficient quality. Two new best VRP solutions were identified through this process.

The success of an adaptive memory procedure is obviously linked to the capacity of the underlying search process to generate a pool of high quality solutions, which is certainly the case of Taillard's[33] TS algorithm. It may not work so well if an unsophisticated heuristic was employed to generate the individual solutions. The use of an adaptive memory certainly adds to computation time, as illustrated by some of the results obtained by Rochat and Taillard[22]. Coding an adaptive memory procedure requires a minimum of computing skills but is not overly complicated. The concept is highly flexible since it can be used in conjunction with other types of heuristics, not only TS, and it can easily be adapted to other contexts. For example, Bozkaya, Erkut and Laporte[60] report an application to the area of political districting.

The granular tabu search algorithm of Toth and Vigo

The idea behind granular tabu search (GTS)[61] is to carry out a preliminary elimination of unpromising arcs or edges from the graph. These are arcs that only have a small likelihood of belonging to an optimal solution. Toth and Vigo[61] eliminate all edges whose cost exceeds a *granularity threshold* $\nu = \beta\bar{c}$, where β is a *sparsification parameter*, and $\bar{c}$ is the average cost of an edge in a good solution identified by means of a fast heuristic. If β is chosen in the interval $[1.0, 2.0]$,then only 10% to 20% of the original edges tend to remain. The value of this parameter is dynamically updated throughout the search process. In their implementation, Toth and Vigo work on the restricted edge set $E(\nu) = \{(v_i, v_j) \in E : c_{ij} \leq \nu\} \cup I$, where I is a set of important edges such as those incident to the depot and those belonging to high quality solutions. The search mechanism implemented by Toth and Vigo uses Taburoute as a subroutine, but tends to produce better results faster.

Because it works on a sparse graph, GTS can quickly perform an extensive search of the solution space. It also produces high quality solutions. The concept of granularity is relatively easy to implement once a good underlying search algorithm is available.

The unified tabu search algorithm of Cordeau, Gendreau, Laporte and Mercier

The unified tabu search algorithm (UTSA) was initially designed by Cordeau, Gendreau and Laporte[5] for the periodic VRP and the multi-depot VRP. It was later slightly modified and extended to the single depot multi-depot and periodic VRP with time windows by Cordeau, Laporte and Mercier[62]. The algorithm shares some features with Taburoute, namely a penalized objective function F' with self-adjusting coefficients to allow the exploration of intermediate infeasible solutions, and the use of continuous diversification, but fixed length tabu durations are used. As in Semet and Taillard[63], the tabu mechanism operates on an attribute set $B(x)$ associated with solution x. More specifically, $B(x) = \{(i,k) :$ vertex v_i is visited by vehicle k in solution $x\}$. Neighbour solutions are obtained by removing an attribute (i,k) from $B(x)$ and replacing it with (i,k'), where $k' \neq k$. The insertion of v_i into route k' is then performed so as to minimize the penalized objective function F', and attribute (i,k) is declared tabu for θ iterations.

The algorithm was highly successful on a variety of VRPs including the classical VRP with capacity and distance restrictions (see Table 3), the periodic and multi-depot VRPs[5], the VRP with site dependencies[4], and the single and multi-depot VRP with time windows[62]. On the classical VRP, it yields solution values within 0.69% of the best known. The average computing time is 13.75 minutes on a Sun Ultrasparc 10 workstation (440 MHz). On all other problems it often identified new best known solutions and has proved superior to several competing methods.

The UTSA ranks high on accuracy and performs rather well on speed, although it cannot compete on this dimension with very fast procedures using simple heuristics such as CW. Since all but one of its parameters are the same for all applications, this heuristic has in effect only one parameter. Moreover, since it operates according to a very simple mechanism, it is probably the simplest of all TS implementations for the VRP, but still it cannot outperform CW in terms of simplicity. Finally, UTSA obtains

excellent marks on flexibility. To our knowledge, this is the only VRP heuristic that can effectively handle so many different variants with a unique set of algorithmic rules and a single user-controlled parameter.

We present in Table 3 comparative computational results for the various VRP metaheuristics just described. Care must be taken when comparing results since the Taillard and adaptive memory solution values are the best over several runs, whereas in the case of Taburoute, GTS and UTSA, all reported values correspond to a single run using standard parameter settings. In the case of Taburoute and UTSA, better solution values (sometimes coresponding to the best) were obtained by either changing parameter values (for Taburoute) or by executing several runs with the same parameters, starting from randomly generated different initial solutions (for UTSA).

Summary and conclusion

In this guide to VRP heuristics we have examined some of the well-known classical methods as well as some of the best available metaheuristics. Contrary to a long established custom of assessing heuristics against accuracy and speed only, we have added what we perceive as two criticial criteria: simplicity and flexibility. The results of our analysis are summarized in Table 4.

None of the classical heuristics fares very well on accuracy and flexibility. In this category, the celebrated Clarke and Wright heuristic has at least the distinct advantage of being very quick and simple to implement. This probably explains its ongoing popularity. However, in contexts where vehicle routes must be planned over a long horizon and large sums of money are at stake, it is worth investing time and money in a method that performs a more extensive exploration of the search space.

Among the tabu search algorithms, there has been a tendancy toward increased speed and simplicity. The adaptive memory procedure and the granularity principle are highly portable concepts that can be used within almost any search process. Among stand-alone heuristics, the unified tabu search algorithm scores very well on

all dimensions.

Acknowledgements

This article was partly written while the third author visited the Centre for Traffic and Transportation at the Technical University of Denmark. This research was partly funded by the Canadian Natural Sciences and Engineering Research Council (NSERC) under grants 227837-00, OGP00338816, OGP0039682, and OGP0036662, and by the Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR) under grant 2002-ER-73080 This support is gratefully acknowledged. The authors thank Edgar Alberto Cabral and François Guertin for their help with programming, and Oli B.G. Madsen who provided some useful references.

References

1. Dantzig, G.B. and Ramser, J.H. (1959). The truck dispatching problem. *Management Science* 6: 80-91.

2. Gendreau, M., Laporte, G., Musaraganyi, C. and Taillard, É.D. (1999). A tabu search heuristic for the heterogeneous fleet vehicle routing problem. *Computers & Operations Research* 26: 1153-1173.

3. Mingozzi, A., Giorgi, S. and Baldacci, R. (1999). An exact method for the vehicle routing problem with backhauls. *Transportation Science* 33: 315-329.

4. Cordeau, J.-F. and Laporte, G. (2001). A tabu search heuristic for the site dependent vehicle routing problem with time windows. *INFOR*. Forthcoming.

5. Cordeau, J.-F., Gendreau, M. and Laporte, G. (1997). A tabu search heuristic for the periodic and multi-depot vehicle routing problems. *Networks* 30: 105-119.

6. Dror, M., Laporte, G and Trudeau, P. (1994). Vehicle routing with split deliveries. *Discrete Applied Mathematics* 50: 239-254.

7. Golden, B.L. and Assad, A.A. (eds) (1988). *Vehicle Routing: Methods and Studies*. North-Holland: Amsterdam.

8. Fisher, M.L. (1995). Vehicle routing. In: *Network Routing, Handbooks in Operations Research and Management Science*, Vol. 8, Ball, M.O., Magnanti, T.L., Monma, C.L. and Nemhauser, G.L. (eds). North-Holland: Amsterdam, pp. 1-33.

9. Desrosiers, J., Dumas, Y., Solomon, M.M. and Soumis, F. (1995). Time constrained routing and scheduling. In: *Network Routing: Handbooks in Operations Research and Management Science*, Vol. 8, Ball, M.O., Magnanti, T.L., Monma, C.L. and Nemhauser, G.L. (eds). North-Holland: Amsterdam, pp. 35-139.

10. Crainic, T.G. and Laporte, G. (eds) (1998). *Fleet Management and Logistics*. Kluwer: Boston.

11. Toth, P. and Vigo, D. (eds) (2001). *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and Applications. Forthcoming.

12. Laporte, G. and Semet, F. (2001). Classical heuristics for the capacitated VRP. In: *The Vehicle Routing Problem*, Toth, P. and Vigo, D. (eds). SIAM Monographs on Discrete Mathematics and Applications. Forthcoming.

13. Gendreau, M., Laporte, G. and Potvin, J.-Y. (2001). Metaheuristics for the capacitated VRP. In: *The Vehicle Routing Problem*, Toth, P. and Vigo, D. (eds). SIAM Monographs on Discrete Mathematics and Applications. Forthcoming.

14. Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M.M. and Soumis, F. (2001). The VRP with time windows. In: *The Vehicle Routing Problem*, Toth, P. and Vigo, D. (eds). SIAM Monographs on Discrete Mathematics and Applications. Forthcoming.

15. Toth, P. and Vigo, D. (1998). Exact solution of the vehicle routing problem. In: Crainic, T.G. and Laporte, G. (eds). *Fleet Management and Logistics*. Kluwer: Boston, pp. 1-31.

16. Clarke, G. and Wright, J.R. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research* 12: 568-581.

17. Gillett, B.E. and Miller, L.R. (1974). A heuristic algorithm for the vehicle dispatch problem. *Operations Research* 22 : 340-349.

18. Fisher, M.L. and Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks* 11: 109-124.

19. Kirkpatrick, S., Gellatt, C.D. Jr. and Vecchi, M.P. (1983). Optimization by simulated annealing. *Science* 220: 671-680.

20. Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research* 13: 533-549.

21. Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press: Ann Arbor, MI.

22. Rochat, Y. and Taillard, É.D. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics* 1: 147-167.

23. Christofides, N., Mingozzi, A. and Toth, P. (1979). The vehicle routing problem. In: *Combinatorial Optimization*, Christofides, N., Mingozzi, A., Toth, P. and Sandi, C. (eds). Wiley: Chichester, pp. 315-338.

24. Barr, R.S., Golden, B.L., Kelly, J.P., Resende, M.G.C. and Stewart, W.R. Jr. (1995). Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics* 1: 9-32.

25. Gendreau, M., Hertz, A., and Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. *Management Science* 40: 1276-1290.

26. Hadjiconstantinou, E.A., Christofides, N. and Mingozzi, A. (1995). A new exact algorithm for the vehicle routing problem based on q-paths and k-shortest paths relaxations. In: Gendreau, M. and Laporte, G. (eds). *Freight Transportation*, Baltzer: Amsterdam, pp 21-43.

27. Mole, R.H (1983). The curse of unintended rounding error: A case from the vehicle scheduling literature. *Journal of the Operational Research* Society 34: 607-613.

28. Ichoua, S., Gendreau, M. and Potvin, J.-Y. (2000). Diversion issues in real-time vehicle dispatching. *Transportation Science* 34: 426-438.

29. Gendreau, M., Laporte, G. and Semet, F. (2001). A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. *Parallel Computing*. Forthcoming.

30. Renaud, J., Boctor, F.F. and Laporte, G. (1996). An improved petal heuristic for the vehicle routing problem. *Journal of the Operational Research Society* 47: 329-336.

31. Golden, B.L., Wasil, E.A., Kelly, J.P. and Chao, I-M. (1998). Metaheuristics in vehicle routing: In: *Fleet Management and Logistics*, Crainic, T.G. and Laporte, G. (eds), Kluwer: Boston, pp. 33-56.

32. Rego, C. and Roucairol, C. (1996). A parallel tabu search algorithm using ejection chains for the vehicle routing problem. In: *Meta-Heuristics: Theory and Application*, Osman, I.H. and Kelly, J.P. (eds), Kluwer: Boston, pp. 661-675.

33. Taillard, É.D. (1993). Parallel iterative search methods for vehicle routing problems. *Networks* 23: 661-673.

34. Solomon, M.M. (1987). Algorithm for the vehicle routing and scheduling problem with time window constraints. *Operations Research* 35: 254-265.

35. Gaskell, T.J. (1967). Bases for vehicle fleet scheduling. *Operational Research Quarterly* 18: 281-295.

36. Yellow, P.(1970). A computational modification to the savings method of vehicle scheduling. *Operational Research Quarterly* 21: 281-283.

37. Nelson, M.D., Nygard, K.E., Griffin, J.H. and Shreve, W.E. (1985). Implementation techniques for the vehicle routing problem. *Computers & Operations Research* 12: 273-283.

38. Paessens, H. (1988). The savings algorithm for the vehicle routing problem. *European Journal of Operational Research* 34: 336-344.

39. Desrochers, M. and Verhoog, T.W. (1989). A matching based savings algorithm for the vehicle routing problem. Les Cahiers du GERAD G-89-04, École des Hautes Études Commerciales, Montreal.

40. Altinkemer, K. and Gavish, B. (1991). Parallel savings based heuristic for the delivery problem. *Operations Research* 39: 456-469.

41. Wark, P. and Holt, J. (1994). A repeated matching heuristic for the vehicle routing problem. *Journal of the Operational Research Society* 45: 1156-1167.

42. Wren, A. (1971). *Computers in Transport Planning and Operation*, Ian Allan: London.

43. Wren, A. and Holliday, A. (1972). Computer scheduling of vehicles from one or more depots to a number of delivery points. *Operational Research Quarterly* 23: 333-344.

44. Foster, B.A. and Ryan, D.M. (1976). An integer programming approach to the vehicle scheduling problem. *Operational Research Quarterly* 27: 307-384.

45. Ryan, D.M., Hjorring, C. and Glover, F. (1993). Extension of the petal method for vehicle routing. *Journal of the Operational Research Society* 44: 289-296.

46. Desrosiers, J., Soumis, F. and Desrochers, M. (1984). Routing with time windows by column generation. *Networks* 14: 545-565.

47. Fisher, M.L., Greenfield, A.J., Jaikumar, R. and Lester, J.T. III (1982). A computerized vehicle routing application. *Interfaces* 12(4): 42-52.

48. Fisher, M.L., Jaikumar, R. and Van Wassenhove, L.N. (1986). A multiplier adjustment method for the generalized assignment problem. *Management Science* 32: 1095-1103.

49. Bramel, J.B. and Simchi-Levi, D. (1995). A location based heuristic for general routing problems. *Operations Research* 43: 649-660.

50. Osman, I.H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research* 41: 421-451.

51. Dueck, G. and Scheuer, T. (1990). Treshold accepting: A general purpose optimization algorithm. *Journal of Computational Physic* 90: 161-175.

52. Dueck, G. (1993). New optimization heuristic: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics* 104: 86-92.

53. Van Breedam, A. (1996). An analysis of the effect of local improvement operators in genetic algorithms and simulated annealing for the vehicle routing problem. RUCA Working Paper 96/14, University of Antwerp, Belgium.

54. Bullnheimer, B., Hartl, R.F., Strauss, C. (1998). Applying the ant system to the vehicle routing problem. In: Voss, S., Martello, S., Osman, I.H., Roucairol, C. (eds), *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer: Boston, pp. 109-120.

55. Ghaziri, H. (1996). Supervision in the self-organizing feature map: Application to the vehicle routing problem. In: Osman I.H. and Kelly, J.P. (eds), *Meta-Heuristics: Theory and Applications*, Kluwer: Boston, pp. 651-660.

56. Willard, J.A.G. (1989). *Vehicle Routing Using r-Optimal Tabu Search*, M.Sc. Dissertation, The Management School, Imperial College, London.

57. Gendreau, M., Hertz, A. and Laporte, G. (1992). New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research* 40: 1086-1094.

58. Taillard, É.D. (1991). Robust taboo search for the quadratic assignment problem. *Parallel Computing* 17: 433-445.

59. Rochat, Y. and Semet, F. (1994). A tabu search approach for delivering pet food and flour in Switzerland. *Journal of the Operational Research Society* 45: 1233-1246.

60. Bozkaya, B., Erkut, E. and Laporte, G. (1999). A tabu search heuristic for political districting. Publication CRT-99-12, Centre for research on transportation, Montreal.

61. Toth, P. and Vigo, D. (1998). The granular tabu search (and its application to the vehicle routing problem). Technical report OR/98/9, DEIS, Università di Bologna, Italy.

62. Cordeau, J.-F., Laporte, G. and Mercier, A. (2001). A unified tabu search

heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society.* Forthcoming.

63. Semet, F. and Taillard, É.D. (1993). Solving real-life vehicle routing problems efficiently using tabu search. *Annals of Operations Research* 41: 469-481.

Table 1 Comparison of five classical heuristics for the VRP

			Clarke and Wright[a]		Two-matching[b]		Sweep[c]		1-Petal[c]		2-Petal[c]		
Instance	n	Type[d]	Value	Seconds[e]	Value	Seconds[f]	Value	Seconds[g]	Value	Seconds[g]	Value	Seconds[g]	Best
1	50	C	578.56	*0.03*	**524.6**[k]	*1200*	531.90	*0.12*	531.90	*0.10*	**524.61**	*0.76*	**524.61**[h,j]
2	75	C	888.04	*0.05*	835.8	*3000*	884.20	*0.17*	885.02	*0.07*	854.09	*0.52*	**835.26**[h]
3	100	C	878.70	*0.10*	830.7	*8700*	846.34	*1.18*	836.34	*0.32*	830.04	*3.84*	**826.14**[h]
4	150	C	1128.24	*0.21*	1038.5	*17100*	1075.38	*2.53*	1070.50	*0.41*	1054.62	*5.93*	**1028.42**[h]
5	199	C	1386.84	*0.32*	1321.3	*28800*	1396.05	*3.60*	1406.84	*0.41*	1354.23	*6.21*	**1291.45**[i]
6	50	C,D	616.66	*0.05*	**555.4**	*1800*	560.08	*0.16*	560.08	*0.09*	560.08	*0.56*	**555.43**[h]
7	75	C,D	974.79	*0.06*	911.8	*2700*	965.51	*0.19*	968.89	*0.07*	922.75	*0.43*	**909.68**[h]
8	100	C,D	968.73	*0.08*	878.0	*9900*	883.56	*1.47*	877.80	*0.25*	877.29	*2.91*	**865.94**[h]
9	150	C,D	1284.63	*0.17*	1176.5	*20700*	1220.71	*3.00*	1220.20	*0.26*	1194.51	*3.58*	**1162.55**[h]
10	199	C,D	1521.94	*0.33*	1418.3	*32100*	1526.64	*4.91*	1515.95	*0.35*	1470.31	*5.19*	**1395.85**[i]
11	120	C	1048.53	*0.14*	1043.4	*16500*	1265.65	*3.52*	1252.84	*0.61*	1109.14	*11.70*	**1042.11**[h]
12	100	C	824.42	*0.08*	**819.6**	*5700*	919.51	*0.64*	824.77	*0.21*	824.77	*2.11*	**819.56**[h]
13	120	C,D	1587.93	*0.12*	1548.3	*30600*	1785.30	*2.24*	1173.69	*0.26*	1585.20	*3.31*	**1541.14**[h]
14	100	C,D	868.50	*0.08*	**866.4**	*8400*	911.81	*0.85*	894.77	*0.17*	885.87	*1.69*	**866.37**[h]
Average deviation from best and time			6.71%	*0.13*	0.63%	*13371.42*	7.09%	*105.6*	5.85%	*15.6*	2.38%	*208.8*	

a. Implemented by Laporte and Semet[12]. Verifiable results (detailed solutions are available in articles or on web sites).
b. Implemented by Wark and Holt[41]. Verifiable results, best of five runs.
c. Implemented by Renaud, Boctor and Laporte[30]. Verifiable results.
d. C: capacity restriction, D: distance restriction.
e. Sun Ultrasparc 10 workstations (42 Mflops).
f. Sun 4/630 MP. Total time for five runs.
g. Sun Sparcstation 2 (210.5 Mips, 4.2 Mflops), 32 megabytes of RAM.
h. Taillard[33]. Verifiable results.
i. Rochat and Taillard[22].
j. Optimal solution value.
k. Bold numbers correspond to best known values.

Table 2. The Fisher and Jaikumar algorithm and the Bramel and Simchi-Levi enhancement

			Fisher and Jaikumar[a]			Location-based heuristic[b]		
Instance	n	Type[c]	Value[d]	Seconds[e]	Best	Value[f]	Seconds[g]	Best
1	50	C	524	*9.3*	*521*[i]	**524.6**	*68*	**524.61**[i,n]
2	75	C	857	*12.0*	*830*[j]	848.2	*406*	**835.26**[j]
3	100	C	833	*17.7*	*815*[i]	832.2	*406*	**826.14**[j]
4	150	C	**1014**[h]	*33.6*	*1014*[k]	1088.6	*2552*	**1028.42**[j]
5	199	C	1420	*40.1*	*1284*[l]	1461.2	*4142*	**1291.45**[m]
6	50	C,D	560	*15.2*	*548*[j]	--	--	**555.43**[j]
7	75	C,D	916	*20.6*	*903*[j]	--	--	**909.68**[j]
8	100	C,D	885	*52.2*	*855*[i]	--	--	**865.94**[j]
9	150	C,D	1230	*121.3*	*1147*[j]	--	--	**1162.55**[j]
10	199	C,D	1518	*136.6*	*1378*[j]	--	--	**1395.85**[m]
11	100	C	--	--	*1035*[i]	1051.5	*1303*	**1042.11**[i]
12	120	C	824	*6.4*	*820*[j]	826.1	*400*	**819.56**[j]
13	100	C,D	--	--	*1529*[j]	--	--	**1541.14**[j]
14	120	C,D	**848**	*6.3*	*848*[k]	--	--	**866.37**[j]
Average deviation from best and time			3.28%	*47.13*		3.29%	*1394.42*	

a. Implemented by Fisher and Jaikumar[18]. Unverifiable results.
b. Implemented by Bramel and Simchi-Levi[49] Verifiable results.
c. C: capacity restrictions, D: distance restrictions.
d. Distances rounded to the nearest integer.
e. DEC-10 computer.
f. Real distances.
g. RS6000, Model 550.
h. Bold numbers correspond to best known values.
i. Solution obtained by means of the Unified Tabu Search Algorithm (UTSA) of Cordeau, Gendreau and Laporte[5], using several runs. Verifiable results.
j Solutions obtained by means of Taburoute[25].Verifiable results.
k. Fisher and Jaikumar[18]. Verifiable solutions of 1017 and 865 were obtained with UTSA for instances 4 and 14.
l. Taillard[33].
m. Rochat and Taillard[22].
n. Optimal solution value[26].

Table 3. Comparison of five metaheuristics for the VRP

			Taburoute[a]		Taillard[b]	Adaptive memory[b]	Granular tabu search[c]		Unified tabu search algorithm[d]		
Instance	n	Type	Value	Minutes[e]	Value	Value	Value	Minutes[f]	Value	Minutes[g]	Best
1	50	C	**524.61**[l]	*6.0*	**524.61**	**524.61**	**524.61**	*0.81*	**524.61**	*4.57*	**524.61**[l,k]
2	75	C	835.77	*53.8*	**835.26**	**835.26**	838.60	*2.21*	835.45	*7.27*	**835.26**[j]
3	100	C	829.45	*18.4*	**826.14**	**826.14**	828.56	*2.39*	829.44	*11.23*	**826.14**[j]
4	150	C	1036.16	*58.8*	**1028.42**	**1028.42**	1033.21	*4.51*	1038.44	*18.72*	**1028.42**[j]
5	199	C	1322.65	*90.9*	1298.79	**1291.45**	1318.25	*7.50*	1305.87	*28.10*	**1291.45**[j]
6	50	C,D	**555.43**	*13.5*	**555.43**	**555.43**	**555.43**	*0.86*	**555.43**	*4.61*	**555.43**[j]
7	75	C,D	913.23	*54.6*	**909.68**	**909.68**	920.72	*2.75*	**909.68**	*7.55*	**909.68**[j]
8	100	C,D	**865.94**	*25.6*	**865.94**	**865.94**	869.48	*2.90*	866.38	*11.17*	**865.94**[j]
9	150	C,D	1177.76	*71.0*	**1162.55**	**1162.55**	1173.12	*5.67*	1171.81	*19.17*	**1162.55**[j]
10	199	C,D	1418.51	*99.8*	1397.94	**1395.85**	1435.74	*9.11*	1415.40	*29.74*	**1395.85**[j]
11	120	C	1073.47	*22.2*	**1042.11**	**1042.11**	1042.87	*3.18*	1074.13	*14.15*	**1042.11**[j]
12	100	C	**819.56**	*16.0*	**819.56**	**819.56**	**819.56**	*1.10*	**819.56**	*10.99*	**819.56**[j]
13	120	C,D	1545.93	*59.2*	**1541.14**	**1541.14**	1545.51	*9.34*	1568.91	*14.53*	**1541.14**[j]
14	100	C,D	**866.37**	*65.7*	**866.37**	**866.37**	**866.37**	*1.41*	866.53	*10.65*	**866.37**[j]
Average deviation from best and time			0.73%	*46.8*	0.06%	0.00%	0.69%	*3.84*	0.69%	*13.75*	

a. Standard algorithm with one set of parameters. Verifiable results.
b. Verifiable results. Best of several runs. Computation times are not reported.
c. Toth and Vigo[61].
d. Solutions obtained by means of the Unified Tabu Search. Algorithm of Cordeau, Gendreau and Laporte[5], without changing the parameters, and with 100,000 iterations.
e. Silicon Graphics workstation (36MHz, 5.7 Mflops).
f. Pentium 200 MHz PC.
g. Sun Ultrasparc 10 (440MHz).
h. C: capacity restrictions, D: distance restrictions.
i. Taillard[33].
j. Rochat and Taillard[22].
k. Optimal solution[26].
l. Bold numbers correspond to best known values.

Table 4. Assessment of some of the main VRP heuristics

Classical heuristics	Accuracy	Speed	Simplicity	Flexibility
Clarke and Wright (CW)[16]	Low	Very high	Very high	Low
Two-matching based methods[41]	High	Very low	Low	Low
Sweep[17]	Low	Medium-High	High	Low
1-Petal[45]	Low	High	Medium	Medium
2-Petal[30]	Medium	Medium	Medium	Medium
Fisher and Jaikumar (FJ)[18]	Difficult to assess	Difficult to assess	Low	Low
Location based FJ by Bramel and Simchi-Levi[49]	Medium	Low	Low	Low
Metaheuristics	**Accuracy**	**Speed**	**Simplicity**	**Flexibility**
Taburoute[25]	High	Medium	Medium	High
Taillard[33]	Very high	Difficult to assess	Medium-Low	High
Adaptive memory[22,a]	Very high	Difficult to assess	Medium-Low	High
Granular tabu search[61,a]	High	Medium	Medium	High
Unified tabu search algorithm[62]	High	Medium	Medium	High

[a] These are not VRP algorithms per se, but mechanisms that can be incorporated within another algorithm to improve its performance. Some of the ratings of these two features are therefore linked to those of the underlying algorithm. They relate to the results presented by Rochat and Taillard[22] and by Toth and Vigo[61] for specific implementations.